ICALEPCS 2025

25 SEPTEMBER 2025
CHICAGO, IL, USA

Better software observability using Tango Controls with OpenTelemetry - experience at MAX IV

Anton Joubert (author)
Marcelo Alcocer (presenter)
Lin Zhu (co-author)
Benjamin Bertrand (co-author)

Agenda

Introduction

Examples

Performance and resources

Conclusion

Introduction

Observability

- Understand complex system from outside
- Logs, traces, metrics, profiles

Introduction

Observability

- Understand complex system from outside
- Logs, traces, metrics, profiles

OpenTelemetry

- Observability framework
- Vendor agnostic
- Open source
- Many programming languages

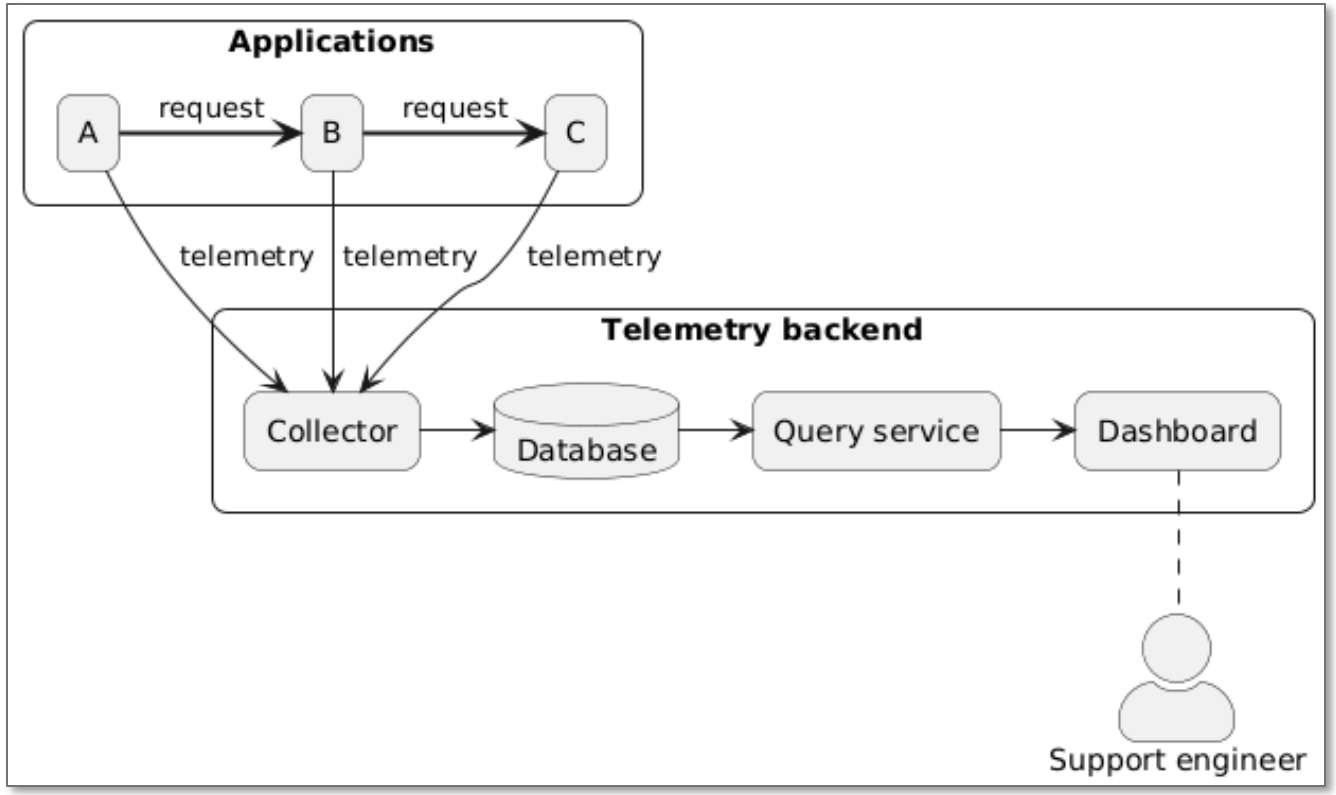
Introduction

Observability

- Understand complex system from outside
- Logs, traces, metrics, profiles

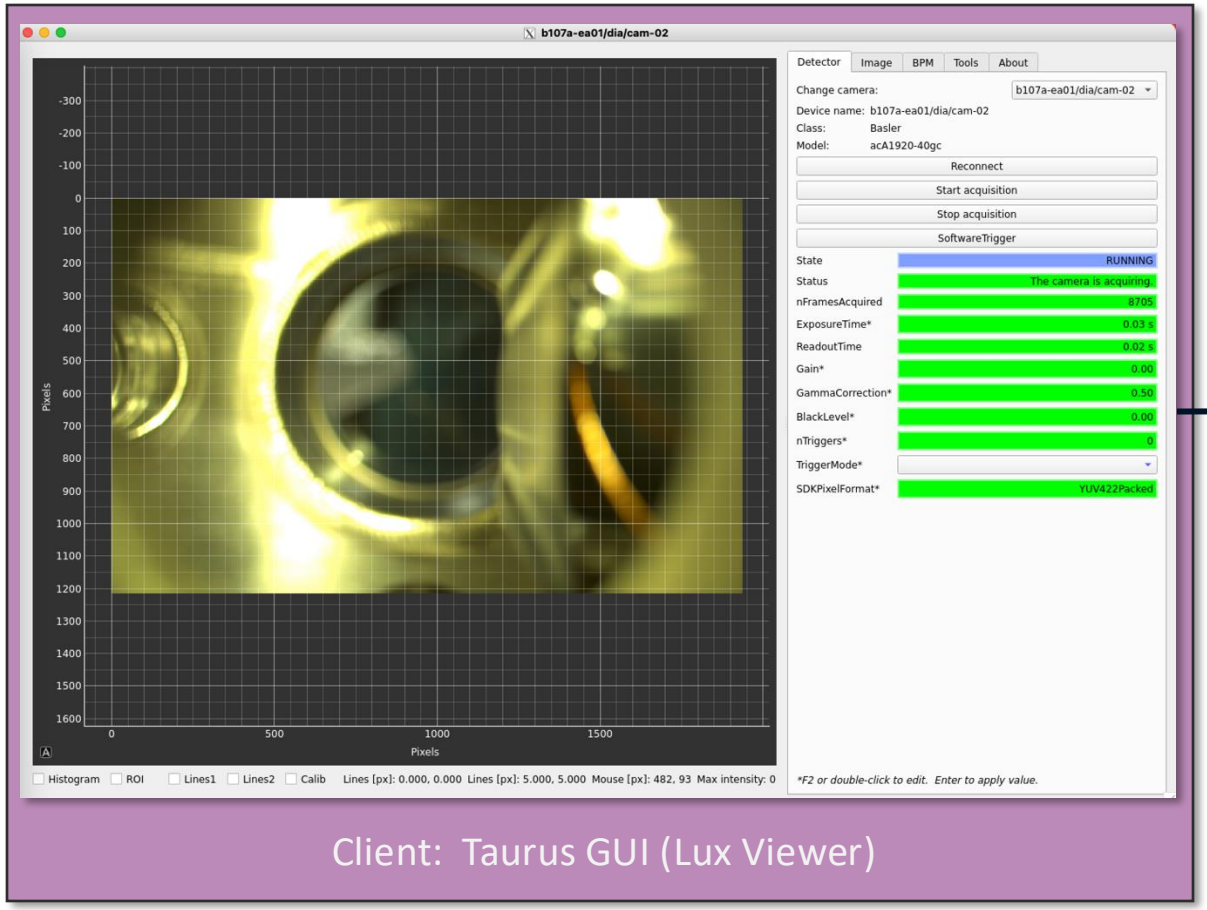
OpenTelemetry

- Observability framework
- Vendor agnostic
- Open source
- Many programming languages

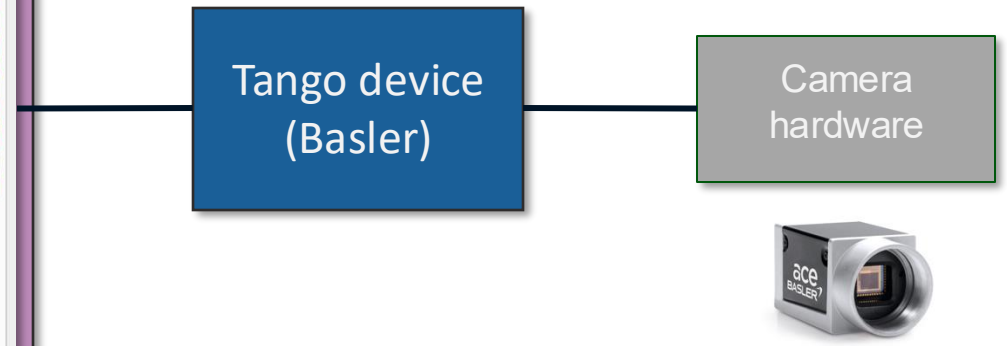


Examples

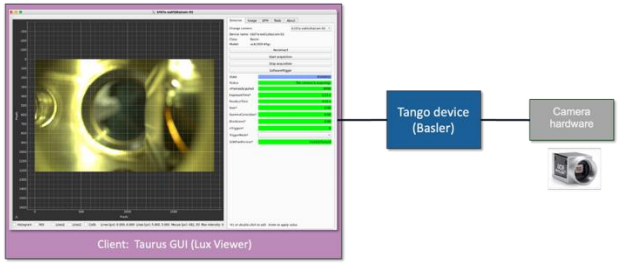
GUI and Tango device for camera



Client: Taurus GUI (Lux Viewer)



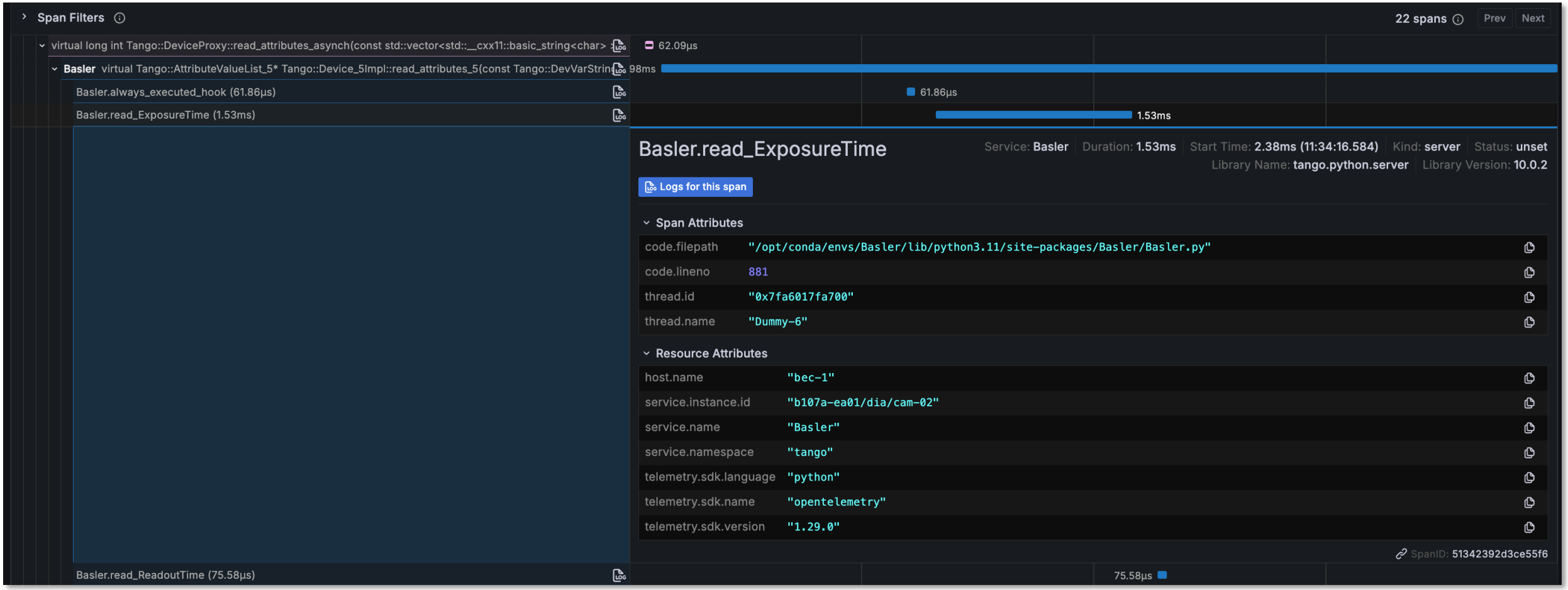
Trace detail



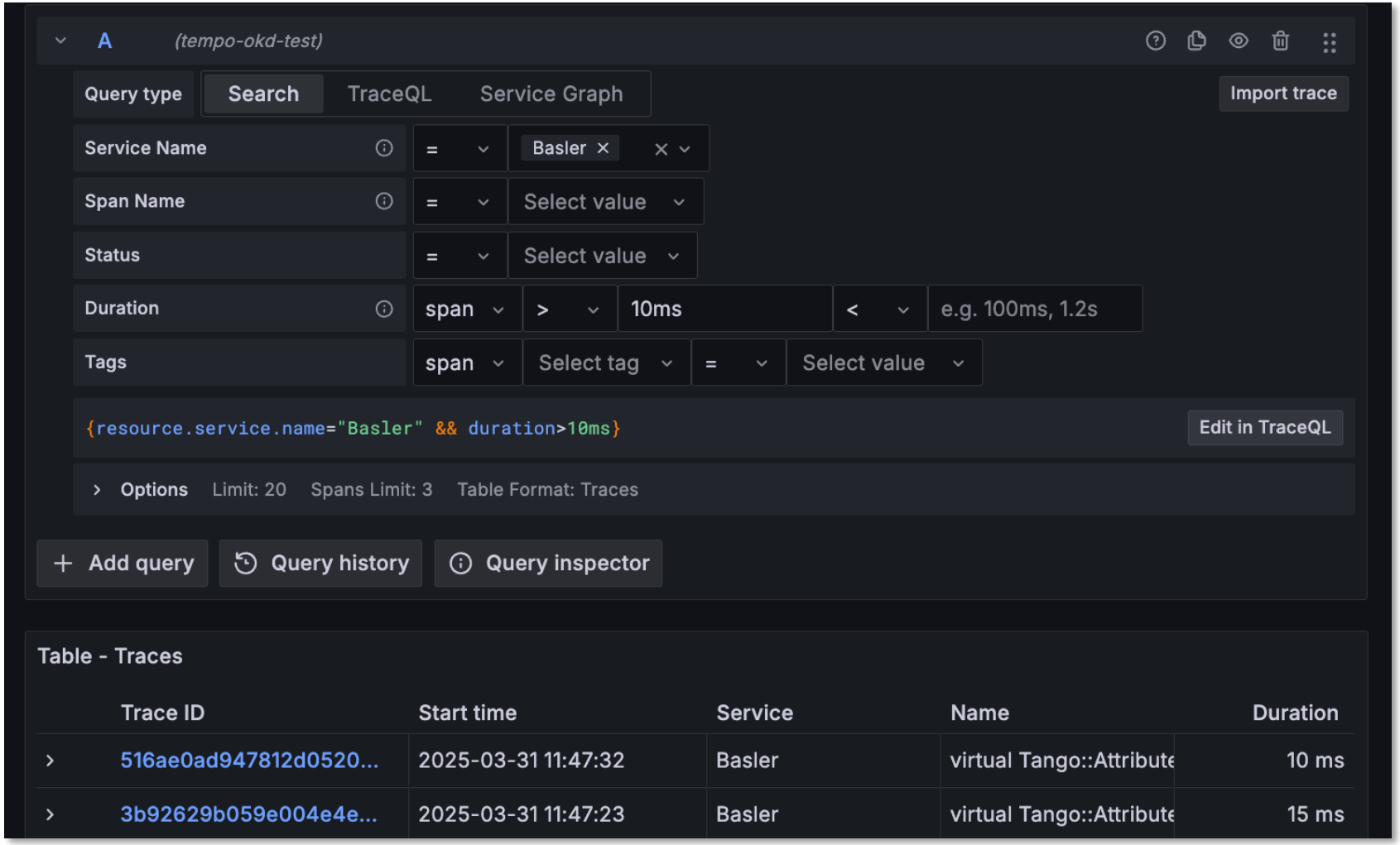
Service & Operation	0µs	1.81ms	3.61ms	5.42ms	7.22ms
pytango.client TangoDevice.__pollAsynch (198.9µs)	198.9µs				
DeviceProxy.read_attributes_asynch (89.94µs)	89.94µs				
virtual long int Tango::DeviceProxy::read_attributes_asynch(const std::vector<std::__cxx11::basic_string<char>	62.09µs				
Basler virtual Tango::AttributeValueList_5* Tango::Device_5Impl::read_attributes_5(const Tango::DevVarString	98ms				
Basler.always_executed_hook (61.86µs)		61.86µs			
Basler.read_ExposureTime (1.53ms)		1.53ms			
Basler.read_ReadoutTime (75.58µs)			75.58µs		
Basler.read_Gain (1.08ms)			1.08ms		
Basler.read_GammaCorrection (50.79µs)				50.79µs	
Basler.read_BlackLevel (38.24µs)				38.24µs	
Basler.read_nTriggers (24.12µs)				24.12µs	
Basler.read_TriggerMode (25.02µs)				25.02µs	
Basler.read_sdkPixelFormat (47.65µs)				47.65µs	
Basler.read_markX (24.08µs)				24.08µs	
Basler.read_markY (21.86µs)				21.86µs	
Basler.read_markX2 (32.09µs)				32.09µs	
Basler.read_markY2 (22.39µs)				22.39µs	
Basler.read_centerX (23.58µs)				23.58µs	
Basler.read_centerY (22.43µs)				22.43µs	
Basler.read_scaleX (21.81µs)				21.81µs	
Basler.read_scaleY (45.6µs)				45.6µs	
Basler.read_Roi (60.74µs)				60.74µs	

User interface: [Grafana Tempo]

Span detail


Span Filters
22 spans
Prev
Next
virtual long int Tango::DeviceProxy::read_attributes_asynch(const std::vector<std::__cxx11::basic_string<char>
62.09µs
Basler virtual Tango::AttributeValueList_5* Tango::Device_5Impl::read_attributes_5(const Tango::DevVarStrin
98ms
Basler.always_executed_hook (61.86µs)
61.86µs
Basler.read_ExposureTime (1.53ms)
1.53ms
Basler.read_ExposureTime
Service: Basler
Duration: 1.53ms
Start Time: 2.38ms (11:34:16.584)
Kind: server
Status: unset
Library Name: tango.python.server
Library Version: 10.0.2
Logs for this span
Span Attributes
code.filepath "/opt/conda/envs/Basler/lib/python3.11/site-packages/Basler/Basler.py"
code.lineno 881
thread.id "0x7fa6017fa700"
thread.name "Dummy-6"
Resource Attributes
host.name "bec-1"
service.instance.id "b107a-ea01/dia/cam-02"
service.name "Basler"
service.namespace "tango"
telemetry.sdk.language "python"
telemetry.sdk.name "opentelemetry"
telemetry.sdk.version "1.29.0"
SpanID: 51342392d3ce55f6
Basler.read_ReadoutTime (75.58µs)
75.58µs

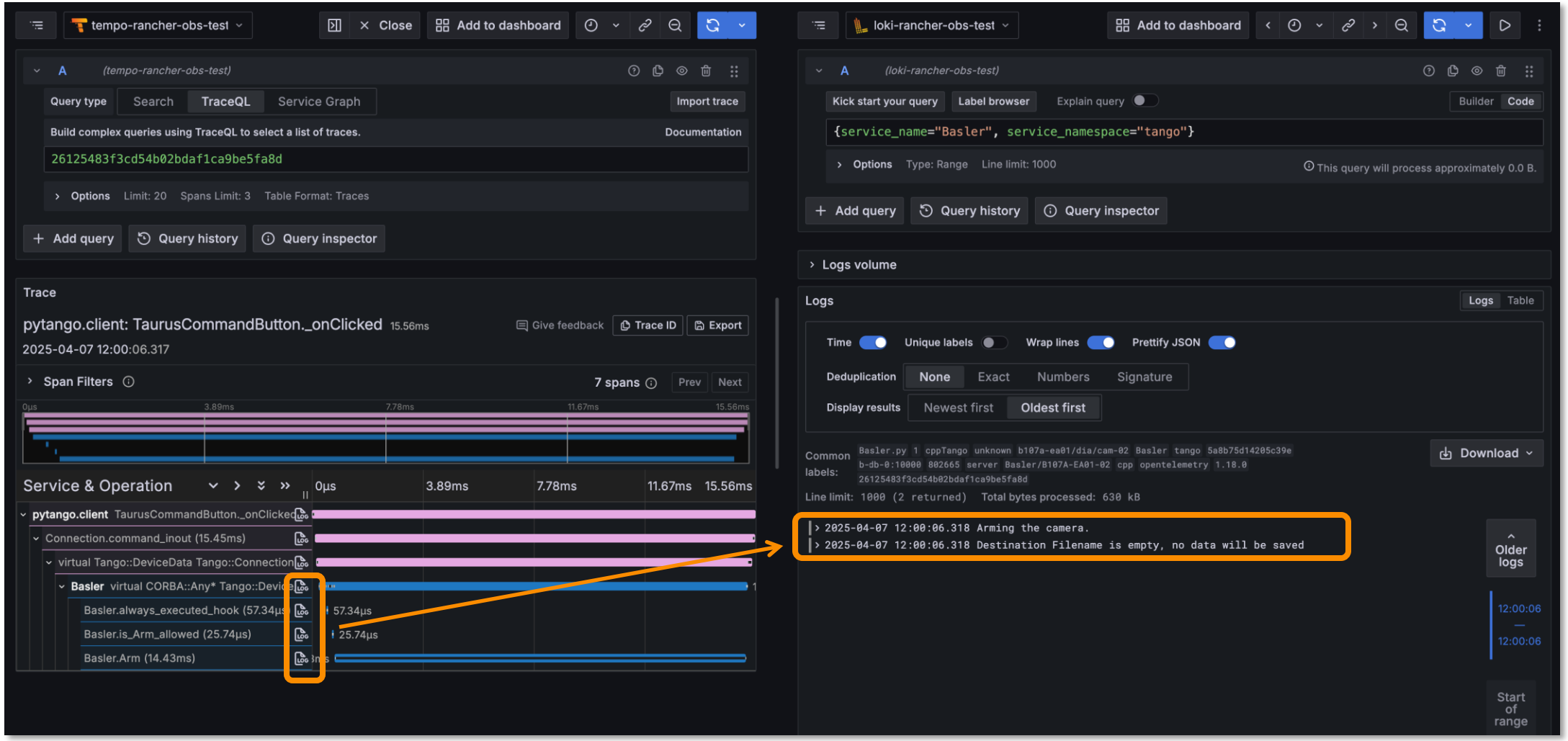
Filtering


A (tempo-okd-test)
Query type
Search
TraceQL
Service Graph
Import trace
Service Name
=
Basler
Span Name
=
Select value
Status
=
Select value
Duration
span
>
10ms
<
e.g. 100ms, 1.2s
Tags
span
Select tag
=
Select value
{resource.service.name="Basler" && duration>10ms}
Edit in TraceQL
Options
Limit: 20
Spans Limit: 3
Table Format: Traces
Add query
Query history
Query inspector
Table - Traces
Trace ID
Start time
Service
Name
Duration
516ae0ad947812d0520...
2025-03-31 11:47:32
Basler
virtual Tango::Attribute
10 ms
3b92629b059e004e4e...
2025-03-31 11:47:23
Basler
virtual Tango::Attribute
15 ms

Slow trace

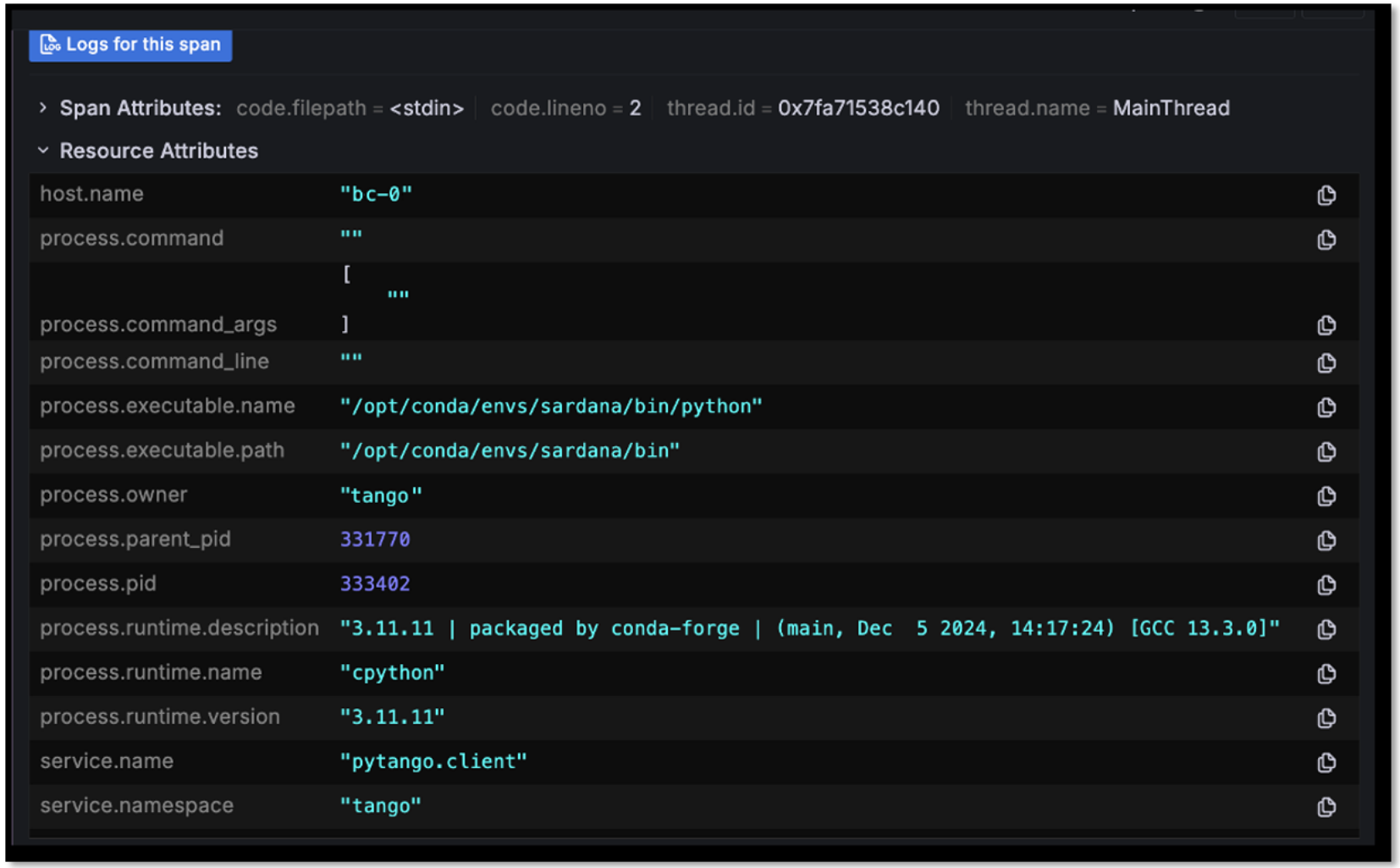

Service & Operation
0µs
3.95ms
7.9ms
11.85ms
15.8ms
Basler virtual Tango::AttributeValueList_5* Tango::De
Basler.always_executed_hook (77.99µs)
77.99µs
Basler.read_ExposureTime (2.95ms)
2.95ms
Basler.read_ReadoutTime (105.63µs)
105.63µs
Basler.read_Gain (1.15ms)
1.15ms
Basler.read_GammaCorrection (104.84µs)
104.84µs
Basler.read_BlackLevel (47.2µs)
47.2µs
Basler.read_nTriggers (29.37µs)
29.37µs
Basler.read_TriggerMode (25.64µs)
25.64µs
Basler.read_sdkPixelFormat (51.24µs)
51.24µs
Basler.read_markX (55.21µs)
55.21µs
Basler.read_markY (22.98µs)
22.98µs
Basler.read_markX2 (22.92µs)
22.92µs
Basler.read_markY2 (22.06µs)
22.06µs
Basler.read_centerX (21.83µs)
21.83µs
Basler.read_centerY (23.12µs)
23.12µs
Basler.read_scaleX (23.1µs)
23.1µs
Basler.read_scaleY (23.07µs)
23.07µs
Basler.read_Roi (64.4µs)
64.4µs

Logs


tempo-rancher-obs-test
Close
Add to dashboard
(tempo-rancher-obs-test)
Query type
Search
TraceQL
Service Graph
Import trace
Build complex queries using TraceQL to select a list of traces.
Documentation
26125483f3cd54b02bdaf1ca9be5fa8d
Options
Limit: 20
Spans Limit: 3
Table Format: Traces
Add query
Query history
Query inspector
Trace
pytango.client: TaurusCommandButton._onClicked
15.56ms
Give feedback
Trace ID
Export
2025-04-07 12:00:06.317
Span Filters
7 spans
Prev
Next
0µs
3.89ms
7.78ms
11.67ms
15.56ms
Service & Operation
pytango.client TaurusCommandButton._onClicked
Connection.command_inout (15.45ms)
virtual Tango::DeviceData Tango::Connection
Basler virtual CORBA::Any* Tango::Devic
Basler.always_executed_hook (57.34µs)
57.34µs
Basler.is_Arm_allowed (25.74µs)
25.74µs
Basler.Arm (14.43ms)
loki-rancher-obs-test
Add to dashboard
(loki-rancher-obs-test)
Kick start your query
Label browser
Explain query
Builder
Code
{service_name="Basler", service_namespace="tango"}
Options
Type: Range
Line limit: 1000
This query will process approximately 0.0 B.
Add query
Query history
Query inspector
Logs volume
Logs
Logs
Table
Time
Unique labels
Wrap lines
Prettify JSON
Deduplication
None
Exact
Numbers
Signature
Display results
Newest first
Oldest first
Common labels:
Basler.py 1 cppTango unknown b107a-ea01/dia/cam-02 Basler tango 5a8b75d14205c39e b-db-0:10000 802665 server Basler/B107A-EA01-02 cpp opentelemetry 1.18.0 26125483f3cd54b02bdaf1ca9be5fa8d
Download
Line limit: 1000 (2 returned)
Total bytes processed: 630 kB
2025-04-07 12:00:06.318 Arming the camera.
2025-04-07 12:00:06.318 Destination Filename is empty, no data will be saved
Older logs
12:00:06
12:00:06
Start of range

Process info

```
$ export OTEL_EXPERIMENTAL_RESOURCE_DETECTORS=process
$ python
```

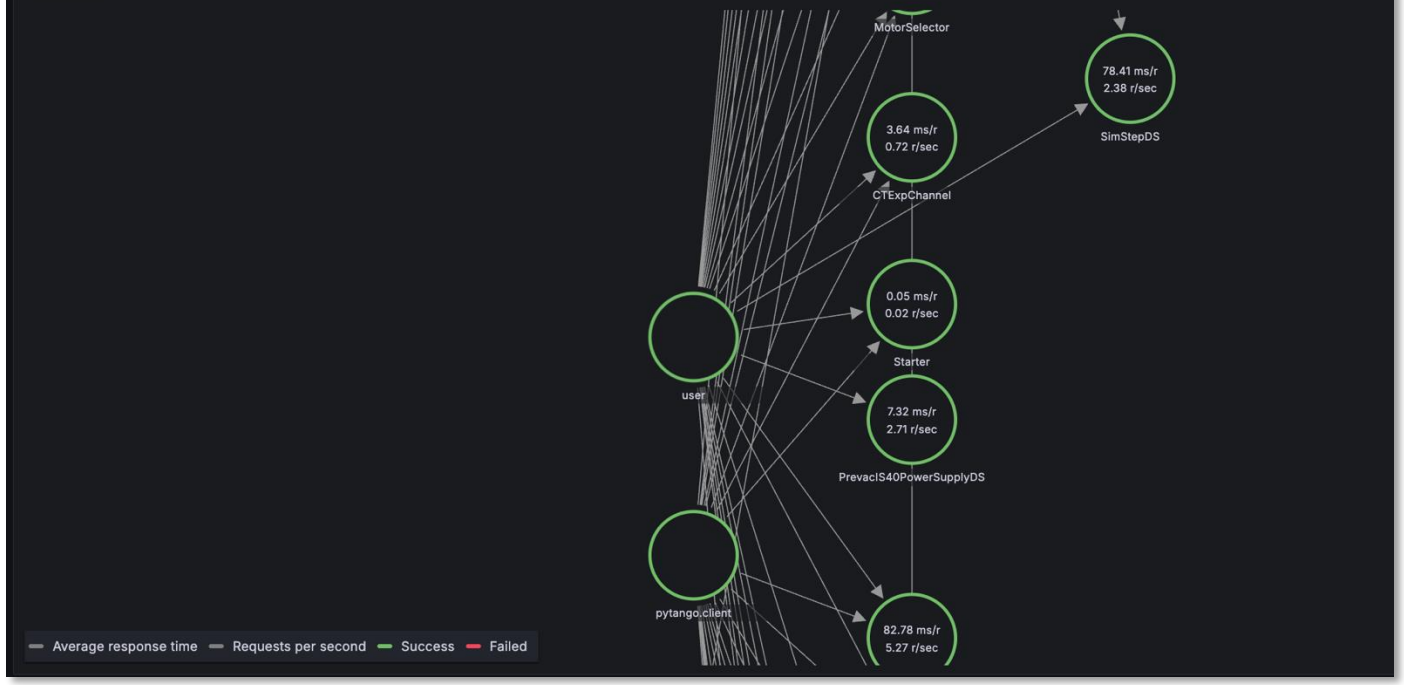


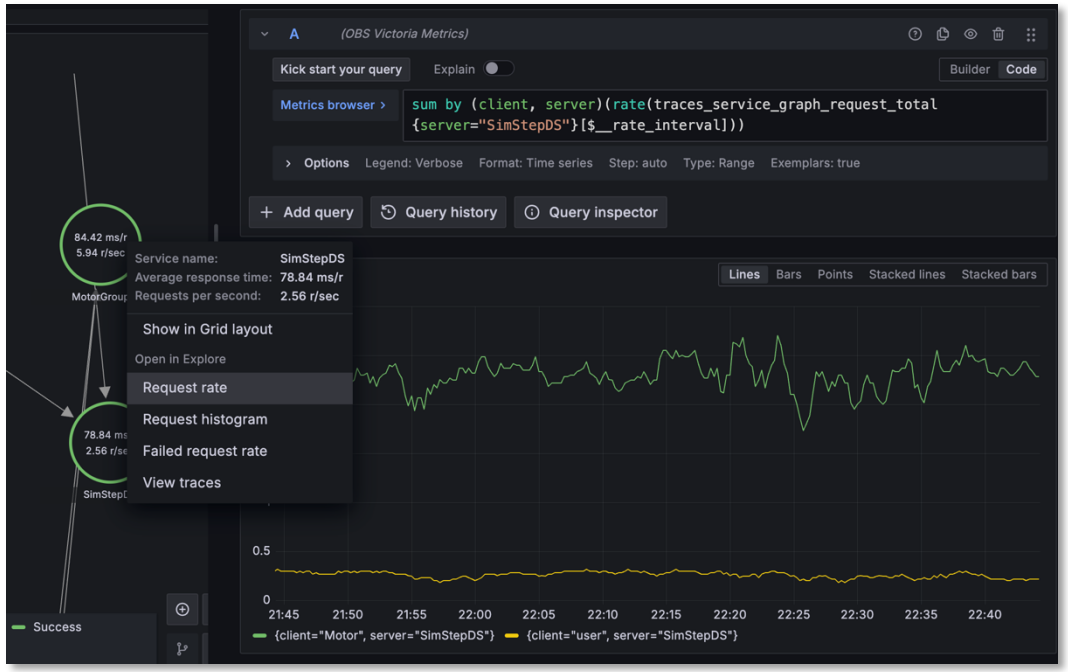
Time series from traces

Service graph

Name	Rate		Error Rate	
virtual Tango::AttributeValueList_5*	423.06	423.059	2.08	2.077
AllenBradleyEIP.read_tag	274.74	274.736	0.00	0.000
AllenBradleyEIP.is_tag_allowed	274.73	274.735	0.00	0.000
AllenBradleyEIP.read_attr_hardware	273.80	273.801	0.00	0.000
virtual CORBA::Any* Tango::Device_	49.74	49.737	0.34	0.340

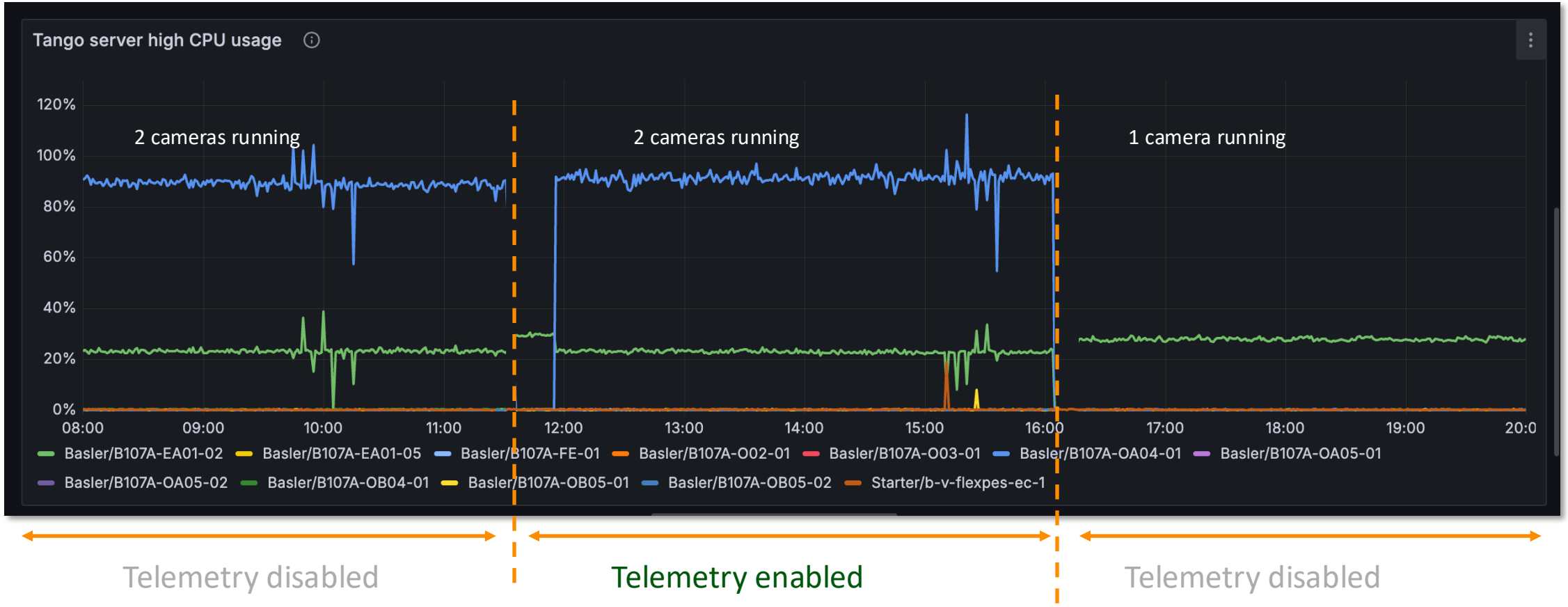
Node graph





Performance and resources

CPU usage (application)



Backend

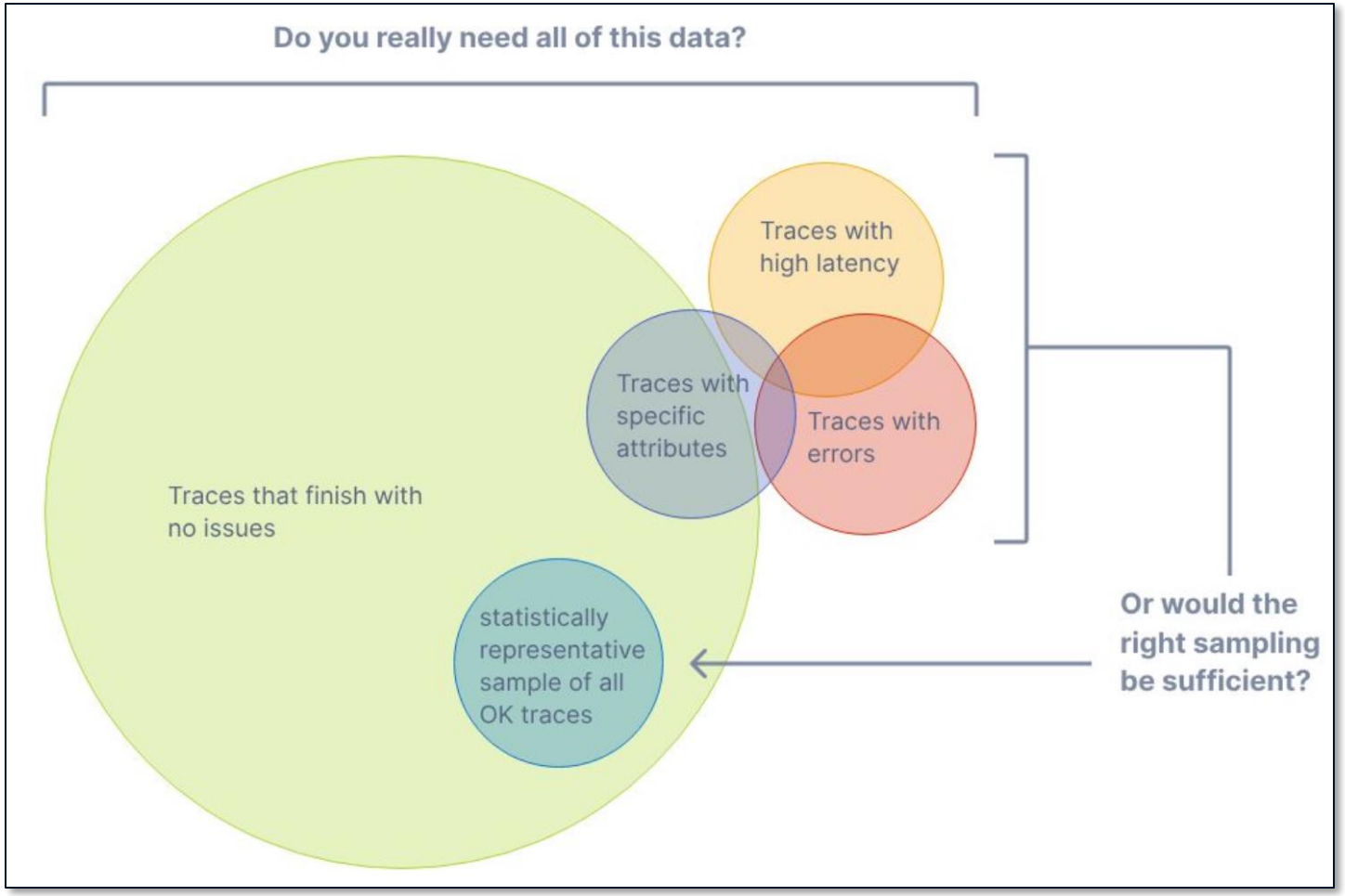
Tested with 1 beamline, 900 Tango devices

Resources (2 replicas each for *Grafana Tempo* and *Loki*)
- 8 CPU cores
- 16 GB RAM

Storage
- 50 GB/day (only keeping 150 GB)
- S3 storage

Network
- 3k spans/second => ~230 HTTP requests/sec

Sampling



Source: https://opentelemetry.io/docs/concepts/sampling/

Conclusion

Conclusion

Very powerful feature in Tango 10

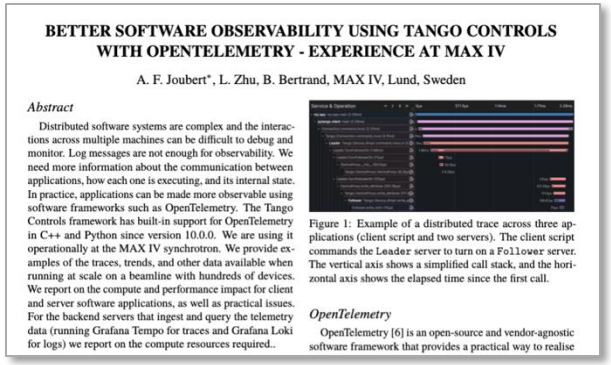
Negligible performance impact on apps

Backend needs “a lot” of compute resources

Trace sampling will be needed to reduce trace volume

Room for improvement

More details in the paper:
WEAG006



BETTER SOFTWARE OBSERVABILITY USING TANGO CONTROLS WITH OPENTELEMETRY - EXPERIENCE AT MAX IV

A. F. Joubert*, L. Zhu, B. Bertrand, MAX IV, Lund, Sweden

Abstract

Distributed software systems are complex and the interactions across multiple machines can be difficult to debug and monitor. Log messages are not enough for observability. We need more information about the communication between applications, how each one is executing, and its internal state. In practice, applications can be made more observable using software frameworks such as OpenTelemetry. The Tango Controls framework has built-in support for OpenTelemetry in C++ and Python since version 10.0.0. We are using it operationally at the MAX IV synchrotron. We provide examples of the traces, trends, and other data available when running at scale on a beamline with hundreds of devices. We report on the compute and performance impact for client and server software applications, as well as practical issues. For the backend servers that ingest and query the telemetry data (running Grafana Tempo for traces and Grafana Loki for logs) we report on the compute resources required..

Figure 1: Example of a distributed trace across three applications (client script and two servers). The client script commands the `Leader` server to turn on a `Follower` server. The vertical axis shows a simplified call stack, and the horizontal axis shows the elapsed time since the first call.

OpenTelemetry

OpenTelemetry [6] is an open-source and vendor-agnostic software framework that provides a practical way to realise

Thank you!

Anton Joubert
anton.joubert@maxiv.lu.se